Knowledge Graph Exploration: A Usability Evaluation of Query Builders for Laypeople

Emil Kuric[1(✉)], Javier D. Fernández[1,2(✉)], and Olha Drozd[1(✉)]

[1] Vienna University of Economics and Business, Vienna, Austria
emil.kuric@s.wu.ac.at, {jfernand,olha.drozd}@wu.ac.at
[2] Complexity Science Hub Vienna, Vienna, Austria

Abstract. SPARQL enables users to access and browse knowledge graphs in a precise way. However, using SPARQL requires knowledge that many casual users lack. To counter this, specific tools have been created that enable more casual users to browse and query results. This paper evaluates and compares the most prominent techniques, QueryVOWL, SPARKLIS and the Wikidata Query Service (WQS), through a usability evaluation, using a mixed-method evaluation based on usability metrics and heuristics, containing both quantitative and qualitative data. The findings show that while WQS achieved the best results, usability problems were encountered in all tools. Key aspects for usability, extracted from the evaluation, serve as important contributions for future query builders.

Keywords: Knowledge graphs · SPARQL · Query builder · Usability

1 Introduction

Linked Open Data (LOD) describes knowledge graphs (KGs) from open sources, in such a way that data can be interlinked [13]. These KGs are ubiquitous, becoming the de facto standard for heterogeneous data integration [5]. Projects such as DBpedia that converts semi-structured content from Wikipedia [2], or Wikidata that offers an open KG created by volunteers [23], are just two KG examples of many successful stories in the Linked Open Data ecosystem.

Traditionally, SPARQL [12], the W3C recommended structured query language for graphs, enables users to access and browse these KGs in a precise way. However, using SPARQL requires a non-negligible knowledge that many end users do not have [9]. While it is a precise and expressive language, it also needs the user to conform to its complex syntax. In addition, it can be difficult to manipulate the interconnected graphs or to gain satisfying results from queries within them [13]. Therefore, SPARQL is mainly geared towards experienced users (i.e., semantic web practitioners) with prior knowledge or insights regarding the SPARQL query language and the structure of datasets (i.e., the

M. Acosta et al. (Eds.): SEMANTiCS 2019, LNCS 11702, pp. 326–342, 2019.
https://doi.org/10.1007/978-3-030-33220-4_24

underlying data model). To counter this, specific tools have been created that enable more casual users (i.e., laypeople) to browse and query results. These so-called SPARQL query builders enable users to generate queries that provide satisfying results either by suggesting relevant parts of the query or through the use of graphical metaphors [9]. While many of these query builders are available, documentation about the evaluation of their usability is scarce. The evaluation process is often not accurately described and therefore may not lead to meaningful results [9].

This paper aims to fill this gap in the scientific literature by evaluating the usability of the most prominent SPARQL query builders. In particular, we focus on studying the usability of SPARQL query builders with users that are inexperienced and have no prior knowledge of SPARQL. For this purpose, we first analyze existing SPARQL query builders and categorize them based on the querying approach into *form-based*, *graph-based* and *natural language-based* query builders. Then, three query builders, the Wikidata Query Service[1] (WQS), QueryVOWL [11] and SPARKLIS [8] are selected based on factors including their availability, querying approach and expressiveness. We compare them using a mixed-method approach consisting of a combination of quantitative and qualitative methods [10]. We first design three tasks to be performed in each of the three tools, and we then conduct a user study with 15 individuals. Quantitative data, that we gather, include the time per task, completion rate and the amount of hints it took to finish the tasks. Furthermore, we evaluate each tool with a System Usability Scale (SUS) questionnaire [6]. Qualitative data are collected through the use of the think-aloud method [7] and the information is analyzed afterwards by clustering the think-aloud protocols with the usability heuristics [19].

Our results show that the querying approach is not as important for the usability of the tool as it may seem, and user satisfaction and preference were mostly influenced by the interface design and ease of use of the tools. In particular, the form-based WQS approach offered the best usability of the three selected tools, although a majority of participants would prefer traditional keyword-based search engines over the presented query builders. We expect that our findings can serve as initial blueprints to guide the next generation of KG query builders.

The rest of the paper is structured as follows: Sect. 2 reviews the functionality and limitations of current SPARQL query builders. Furthermore, we provide details of the three selected tools for the usability evaluation, and explanations as to why they were selected. Section 3 shows the design of our mixed-method approach. Then, in Sect. 4, we analyze the results of our user study and provide lessons learned summarizing the insights gathered and making suggestions for future query builder tools. Finally, Sect. 5 provides conclusion and future work.

2 SPARQL Query Builders

SPARQL query builders are tools that are specifically designed to facilitate the process of querying. A range of these tools are available, each with their own

[1] https://query.wikidata.org/.

varying purposes, querying approaches and target audiences [9]. In this section, we first categorize and analyze available tools. Then we select the most prominent tools for our usability evaluation, based on comparative criteria.

2.1 Categorization

Query builders can be categorized and differentiated based on many possible criteria. In this paper, we follow a pure user interface (UI) criteria influenced by Grafkin and Mironov [9], suggesting a categorization on the basis of the querying approach used in the query builders. Thus, we distinguish between *form-based*, *graph-based* and *natural language-based* querying approaches. It is important to highlight that some approaches combine different elements, following a "hybrid" approach [9]. Other existing techniques that can complement these approaches, such as AutoSPARQL [17], which uses (supervised) machine learning, are outside of the scope of this paper.

Form-Based Query Builders. Form-based querying is an approach that focuses on textual input fields and constructs a query one step at a time. The approach resembles SPARQL's triple pattern syntax and eases the process of query building. This is either enabled by suggesting relevant parts of the query to users or restricting them to selective inputs. A limitation of this approach is that it only allows for a limited set of queries. The classes and objects have to be suggested or enabled as a choice for the user to be able to use them in the query building process. Furthermore, these tools often do not allow for the specification of filters for the results, as they are limited by their input fields. Therefore, some tools are not able to formulate advanced queries [17].

Examples of form-based approaches are ExConQuer [1], the Linked Data Query Wizard [14], VizQuery[2] and the Wikidata Query Service. The ExConQuer Framework is a set of tools meant to explore, convert, and query linked data. In the ExConQuer query builder, users can navigate through classes, instances and properties in a way similar to facet-based browsing. It was deemed as useful both by experts and casual users for exploring and querying linked data [1]. However, it does not offer the full expressiveness of SPARQL.

The Linked Data Query Wizard [14] (LDQW) was designed as a web-based tool for exploring, filtering and analyzing data from SPARQL endpoints. Its approach is to turn the underlying graph structure into a tabular interface that enables interaction with the data set. This interface is meant to take advantage of the fact that many users are already familiar with search engines and spreadsheet applications. A user study conducted by the creators of the tool showed that it was very usable. However, users had difficulties with adding filters and the spreadsheet approach showing too many options [14].

The VizQuery tool is based on SPARQL triple patterns and used specifically for querying data from the Wikidata endpoint. It is a prototype that only offers

[2] https://tools.wmflabs.org/hay/vizquery/.

the simplest functionality for creating queries. VizQuery uses the Wikidata API to provide auto-completion and suggestions for Wikidata properties and items. Although it offers the functionality to use variables for more advanced users, the UI is very limited in its approach as it does not allow for the creation of more complex queries.

Finally, the Wikidata Query Service (WQS) is the official query service of Wikidata and offers a "Query Helper" that allows users to create queries through a form-based approach. As shown in Fig. 1, users can add items to the filter and get relevant suggestions and auto-completion for properties. The Wikidata API allows users to create queries even if the exact names for entities are unknown. Additionally, if users add an item to the filter, WQS will automatically assign a relevant property. However, the WQS does not offer the full expressiveness of SPARQL, and it is meant to be domain-specific as it can only reach Wikidata.

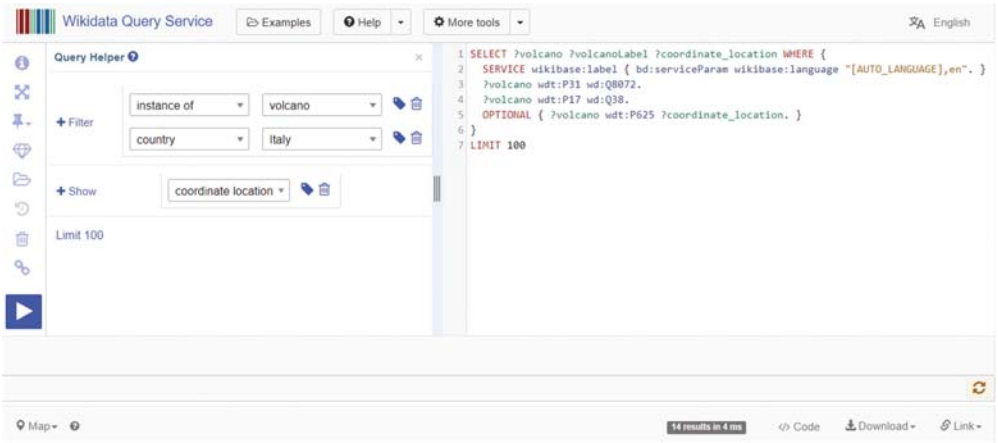

Fig. 1. Screenshot of WQS query builder

Graph-Based Query Builders. The graph-based approach consists of visual query builders and systems (VQS). This category describes tools that lower the difficulty of creating SPARQL queries by enabling a visual approach. The used visualizations are similar to the syntax of textual SPARQL queries. VQS supports users in creating syntactically valid queries by constraining and guiding their editing actions through the use of a graphical UI. A limitation of this approach is that users of these tools still need a rough understanding of the underlying schema to formulate a query. Without understanding how SPARQL queries are constructed users are not able to successfully visualize the queries in some tools [17].

Examples of VQS include iSPARQL[3], NITELIGHT [21], OptiqueVQS [22] and QueryVOWL [11]. iSPARQL and NITELIGHT allow for the whole expressiveness of SPARQL and are query builders for advanced users. Both tools extend the traditional SPARQL framework to enhance its functionalities and feature drag and drop interfaces to connect graph nodes and predicates. However, they

[3] http://dbpedia.org/isparql/.

use a complex series of buttons and options to incorporate all their features in the interface. Additionally, because of their complexity they require thorough knowledge of the underlying data [21].

OptiqueVQS [22] is primarily meant to be a product for end users with limited technical skills and knowledge. Therefore, it includes a simplified interface that is meant to enable users to address basic tasks. A criterion that differentiates OptiqueVQS from other query builders is that it was developed to meet industrial requirements and was evaluated with industrial users. It was, therefore, designed to provide a good balance between usability and expressiveness [22], although, it puts the focus on a very concrete profile of users.

Finally, QueryVOWL [11] differs from other graph-based tools because it uses the Visual Notation for OWL Ontologies (VOWL) [18] to visualize the queries. QueryVOWL enables casual users to build queries by combining the proven to be intuitive and understandable VOWL with matching SPARQL mappings. It offers a drag and drop enabled graphical UI, where users insert nodes through a search box and connect them with predicates or other nodes (see Fig. 2). While QueryVOWL supports most of the expressiveness of SPARQL, it is still somewhat limited in its node-based approach with missing functions and bugs, inherent characteristics of an initial prototype [11].

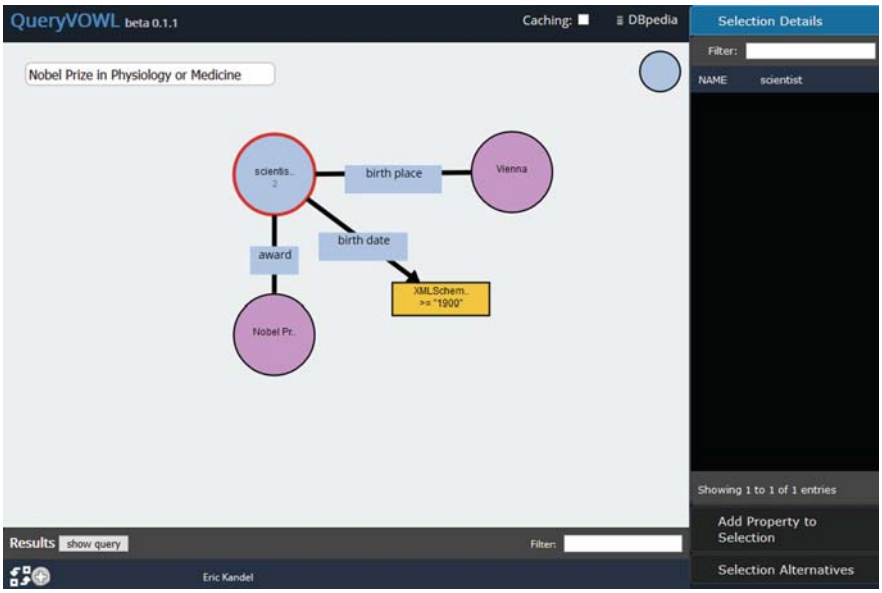

Fig. 2. Screenshot of QueryVOWL query builder

Natural Language-Based Query Builders. Natural language (NL) approaches offer users the convenient and valuable option of using natural language for querying. NL-based approaches enable the user to form precise queries by providing the high expressiveness of terms that users are familiar with. As NL-based approaches interpret natural language phrases, linguistic considerations have to be taken into account. Tools of this variety are often limited by linguistic ambiguities and variability. Furthermore, the development of accurate NL interfaces is complex and requires considerable implementation efforts.

Therefore, NL tools are often domain-specific or tailored to applications making them hardly adaptable to other ontologies [15].

Examples of NL-based tools are NLP-Reduce [16] and SPARKLIS [8]. The NLP-Reduce tool is meant to facilitate the querying of linked data for users with no prior knowledge by using a reduced set of natural language processing functionalities. The system allows for a non-restrictive query language that can consist of keywords, sentence fragments or full sentences. NLP-Reduce was deemed as easy to use without any training. However, as the simple approach avoids complex linguistic and semantic technology it does not allow for the expressiveness of SPARQL [15].

SPARKLIS [8] is a natural language-based web tool that offers the full expressiveness of SPARQL and is usable for casual users. As shown in Fig. 3, the query in SPARKLIS is represented as a NL sentence in a tree structure and the user can focus on different parts of the query to refine it. The selection of parts is guided by suggestions that are enabled by SPARKLIS to allow the user to see relevant options. If a query element is inserted at the focus, the NL sentence is verbalized into a readable form to adapt to that change. Because of its navigational approach and the way suggestions are generated, SPARKLIS has some problems regarding loading and response times [8].

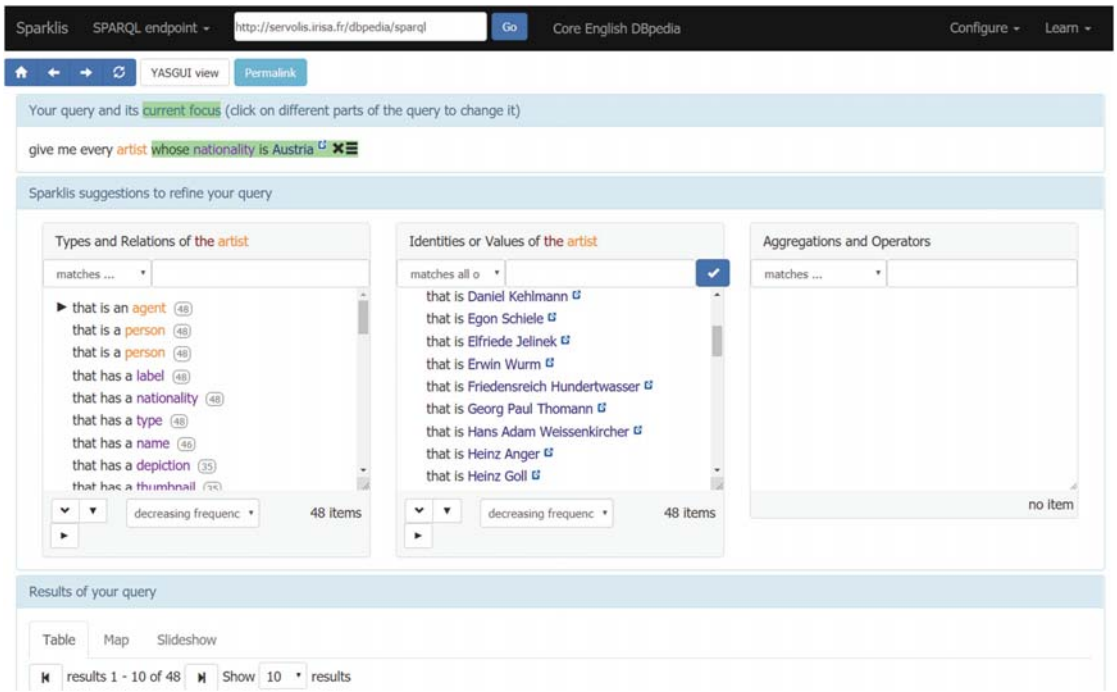

Fig. 3. Screenshot of SPARKLIS query builder

2.2 Selected Tools for Our User Study

We select representative tools for our study, based on criteria, summarized in Table 1. One of the most important factors of the selection was the *availability* of tools. As many tools were either not available or operational anymore,

such as ExConQuer or NITELIGHT, they could not be evaluated. Web tools were preferred because casual users could theoretically find them on the web by themselves and use them to query data. The ease of use or *focus on laypeople* was important as a pre-selection criteria[4], to enable the evaluation with casual users. We considered a tool with a focus on laypeople, (i) if it was specified as designed for casual users in the companion research paper(s) describing the tool, or (ii) if initial examination tests with laypeople revealed a considerable facility when performing simple tasks.

Thus, our user study finally considers WQS, QueryVOWL and SPARKLIS as prominent (and available) query builder representatives of form-, graph- and NL-based query approaches, respectively. Out of all three tools the WQS offers the lowest amount of expressiveness and is furthermore the only tool meant to be domain-specific as it can only reach Wikidata. QueryVOWL has an intuitive graph-based web tool and allows for a certain extent of expressiveness of the SPARQL language. Finally, out of all three selected tools, SPARKLIS offers the most features of SPARQL and therefore the highest expressiveness, although the overload of information and options can negatively influence its usability. The following sections provide an extensive usability evaluation of the three selected query builders: WQS, QueryVOWL and SPARKLIS.

Table 1. Examined tools with selection criteria and rating (✓ = yes, ∼ = partially, - = no). Selected tools are marked in bold

Query builder	Category	Availability	Focus on laypeople	Expressiveness
ExConQuer	form-based	-	✓	∼
LDQW	form-based	✓	∼	-
VizQuery	form-based	✓	✓	-
WQS	form-based	✓	✓	∼
iSPARQL	graph-based	✓	-	✓
NITELIGHT	graph-based	-	-	✓
OptiqueVQS	graph-based	-	✓	-
QueryVOWL	graph-based	✓	✓	∼
NLP-Reduce	NL-based	-	✓	-
SPARKLIS	NL-based	✓	∼	✓

3 Evaluation Design

Our usability evaluation follows a mixed-method evaluation design, consisting of both a quantitative and qualitative part. It is meant to use the results of one method to clarify the results of the other method and therefore to increase

[4] Note that the final usability is evaluated in the next section, here we just select the representative tools of each category.

the meaningfulness of results by capitalizing on the inherent strengths of both methods. This ensures the comparability of quantitative data while it also allows for the informative value found in qualitative approaches [10]. We first show the quantitative and qualitative data considered in our study. Then, we present the participants, test plan and tasks in our study.

3.1 Quantitative Data

For the quantitative data, a suitable conceptual model has to offer measures that can be collected and analyzed in an easy way, while still being meaningful and usable for the evaluation of query builders. In our evaluation, the decision falls on the software engineering standard ISO 9241-11 [4]. In particular, we consider a combination of ISO 9241-11 factors with measurable attributes, as follows. First, we use the *effectiveness* factor, i.e., the success in achieving goals. For the context of the study this factor is further decomposed into the *accuracy* (i.e., amount of hints given to the user during task completion) and *completeness* (i.e., task completion rate) of the tasks performed by the users. Then, we consider the *efficiency*, which describes the amount of resources that users spend to achieve their goals [4] and is measured by the time spent to complete a task. Finally, we use the *satisfaction* factor, i.e., the user's positive attitude towards the tool [4]. In our evaluation, we make use of the System Usability Scale (SUS) questionnaire[5] [6], to provide a quantitative measurement about the users' perceived usability of the tool, facilitating a comparison.

3.2 Qualitative Data: Think-Aloud

Qualitative data that are collected in usability evaluations typically consist of observational findings about the usability of design features. We make use of the think-aloud method [7], i.e., users are asked to voice their thoughts while trying to solve a predefined task. Their thoughts are then gathered in the form of a think-aloud protocol. This method has proven to be a reliable source of information, yet its application offers some challenges. In a realistic scenario, some users will have problems with voicing their inner speech. To counter this, we include a brief explanation of think-aloud in the pretest introduction and *prompt* participants to voice their thoughts during task completion. The think-aloud protocol on its own would be too inaccurate as it is often missing thought processes that are not verbalized. Thus, *retrospective questioning* is used in the evaluation, directly after testing the query builders, to insure that participants are still able to remember their thoughts. Participants are asked to recall their thoughts and opinions on certain points in an unstructured interview.

Typically, experts would evaluate systems or tools on the basis of so-called usability heuristics [19] and through that be able to find problems in UIs. We follow this approach in our evaluation, and analyze the think-aloud protocols based on the ten usability heuristics by Nielsen [19] that provide general principles for

[5] Available at: https://bit.ly/2YuQyHJ.

the design of UIs. The heuristics observed in our evaluation are: *visibility of system status* (i.e., keeping users informed of the process), *match between system and real world* (i.e., speaking the user's language), *user control and freedom* (i.e., enabling users to control their workflow and undo and redo actions), *consistency and standards*, *error prevention*, *recognition rather than recall* (i.e., minimizing the user's memory load), *aesthetic and minimalist design*, and *help and documentation*.

3.3 Participants

While an evaluation of systems such as query builders is typically carried out by field experts, our test users have to have no prior knowledge regarding LOD or SPARQL, so that the usability for casual users can be analyzed. Thus, our selected target group is digital natives that are versed in dealing with web tools and interfaces.

We follow Nielsen's studies on UI evaluations, estimating that the optimal number of participants for a medium-large project should include 15 users [20]. Thus, 15 bachelor students, ranging from 18 to 25 years old and evenly split between genders are finally selected to participate in our usability evaluation. The participants had never heard of or worked with SPARQL.

3.4 Usability Evaluation

The evaluation took approximately an hour per person and consisted of three parts: (i) the introduction that explained the test procedure to the participant, including the topic and evaluation approach of the user study; (ii) the testing of the query builders and the corresponding tasks; (iii) a debriefing where participants were able to verify their think-aloud messages and to add something to their protocols based on the ability to compare all tools. The order of the three tools was randomized before the evaluation. The tasks were printed out on a sheet of paper, and read aloud by the participants so that any questions could be clarified before users started the evaluation.

The participants received the following tasks: (i) show all Austrian artists; (ii) show all volcanoes in Italy and their location (on a map); (iii) show all scientists born in Vienna after 1900 that have been awarded a Nobel Prize in Physiology and Medicine. The first task introduces the participants to the query builders and encourages them to interact with the interface and create a list of results. The second task is similar to the first one in that it combines a single object with one relation to a subject, however, it introduces the location. As QueryVOWL does not offer an option to show the location on a map, the task is changed to output the location attribute. The third task is the most complex one as it is meant to show the expressiveness of the tools. It includes three objects that have to be put into relation with one another and a filter based on the birth date. For WQS the task was changed to search for humans instead of scientists, to output the items, as the corresponding results are saved differently in Wikidata. Furthermore, because the tool is missing a function to filter the results, the WQS

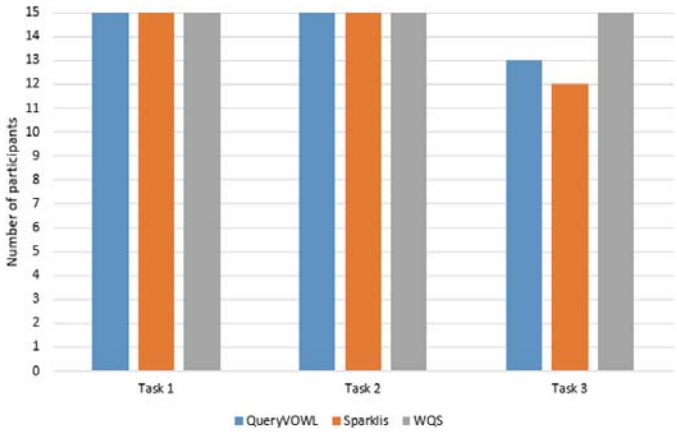

Fig. 4. Task completion (over a total of 15 participants)

Table 2. Average hints per query builder and task

Query builder	Task 1	Task 2	Task 3	Total hints
WQS	**1.07**	0.87	**0.67**	**2.60**
QueryVOWL	1.93	**0.60**	1.73	4.27
SPARKLIS	1.80	1.20	1.53	4.53

task was changed to additionally output the birth year instead. Hints that were given to users in the tasks were mostly based on the way the relations or filters were created. Users were never told where to click but were instead made aware that they had made an error and that their current approach would not work. If users were not sure about which relation they had to choose, a keyword was given to them, which was not counted as a hint. After finishing the tasks for one tool the participant fill out the SUS questionnaire and shortly answer the retrospective questions for the think-aloud protocol.

4 Usability Results

In the following, we present the results[6] of the user study divided into the usability metrics and heuristics that were gathered through the quantitative and qualitative parts of the user study respectively. We then provide a discussion assessing the selected tools.

4.1 Usability Metrics

The usability metrics are analyzed by comparing the quantitative data of the corresponding query builders.

Effectiveness. The effectiveness is comprised of the completeness and the accuracy. Regarding the completeness, most of the participants were able to complete

[6] All fine-grained results are available at: https://bit.ly/2YuQyHJ.

all the tasks for each query builder, as shown in Fig. 4. In particular, the same two participants failed at task three for both QueryVOWL and SPARKLIS, and one participant only failed at the third task for SPARKLIS.

In turn, Table 2 reports the accuracy results, measured by the average number of hints (per participant) that they needed for each task and query builder. Results show that, on average, participants required the most amount of hints to finish the first task. This typically corresponds to the familiarization with the environment. Overall, WQS needed the least amount of hints, about half as much as QueryVOWL and SPARKLIS.

Interestingly, while QueryVOWL excels at the second task (i.e., users seemed to accurately understand and apply similar notions of the first task) it provides the worst results for the most complex third task. Finally, it is worth mentioning that no participant was able to finish all the tasks of a query builder without at least one hint.

Both completeness and accuracy show that WQS offers the best effectiveness. It required the least amount of total hints and achieved the best task completion rate. The effectiveness of QueryVOWL and SPARKLIS showed only marginal differences.

Efficiency. The efficiency was measured by the amount of time that was spent to complete a task. The results in Table 3 show that, as expected, participants needed the most time to finish task three (i.e., the most complex one). In turn, users required the least amount of time to finish the second task, because of the aforementioned learning effect and its similarity to the first task. Overall, efficiency results are in line with the accuracy. Thus, users spend the least amount of time per task in WQS, while the results of QueryVOWL and SPARKLIS are similar to each other.

System Usability Scale. As mentioned in Sect. 3, we measure satisfaction through the System Usability Scale (SUS) [6], in the range 0–100. The results in

Table 3. Average time spent per query builder and task (in mm:ss)

Query builder	Task 1	Task 2	Task 3	Total time
WQS	**01:31**	**01:45**	**01:52**	**05:09**
QueryVOWL	02:42	01:53	03:08	07:44
SPARKLIS	02:26	01:48	03:06	07:21

Table 4. Average System Usability Scale (SUS) score per query builder

Query builder	SUS score	Rating
WQS	**61**	'OK'
QueryVOWL	50.5	'Poor'
SPARKLIS	48.5	'Poor'

Table 4 show relatively similar scores for each tool. WQS reports the highest SUS score of 61, which can be interpreted as an 'OK' result using the adjective rating of the scale [3]. In contrast, the similar scores of QueryVOWL and SPARKLIS can both be rated as 'Poor'.

4.2 Usability Heuristics

The think-aloud protocols were clustered via the so-called usability heuristics [19] (see Sect. 3). We exclude the heuristics that were missing in the protocols and were not directly observed. In the following, we briefly summarize the results that were most frequently mentioned.

Most QueryVOWL users complained about a lack of **visibility of system status**. The graph-based construction was lacking appropriate feedback in some cases and users were not able to see that input has been received. The NL-based SPARKLIS approach had less problems with the visibility, as feedback was provided instantly in NL form. However, users complained about the loading and response times of the tool.

As for the **match between system and real world**, some users of WQS had problems deciphering the meaning of the used terms. As WQS only offered an option to add items as well as properties through the use of the filter button, it resulted in some mistakes for the users. The terms that were used in QueryVOWL were taken directly from SPARQL and were confusing for some of the users. For example, users had problems deciding between using 'artist' as a class, individual or property, because they had no idea of the underlying data model. SPARKLIS managed to speak the user's language through the use of non-system-oriented terms and a natural language query.

Regarding the **user control and freedom**, most users complained about the missing undo and redo buttons for WQS and QueryVOWL. SPARKLIS offered the best user control (e.g., undo and redo) and freedom for participants.

As for **consistency and standards**, WQS and SPARKLIS offered standard buttons and users were able to use them easily. In contrast, none of the users was familiar with the visual style of QueryVOWL.

Regarding **error prevention**, all tools offered suggestions, enabling users to pick from available options. However, WQS was the only tool that offered spell checking and suggestions based on the input, a feature that users were missing in the other tools. For example, if 'geolocation' was entered, it still showed 'coordinate location'.

Concerning **recognition rather than recall**, users described some options of WQS as hidden away. Users complained about QueryVOWLs options, in the nodes and sidebar, which were not selectable and visible in some cases.

As for the **aesthetic and minimalist design**, most users complained about the clutter of information in parts of the interface of WQS, such as the suggestion box. QueryVOWL was described as minimalist to the point of missing necessary information. Users complained about the amount of options and features that were shown at once in SPARKLIS. This led to them feeling overwhelmed and not in control at first.

Table 5. Summary of usability results per tool (bold values show the best tool for each metric)

Query builder	Completion rate (%)	Average hints	Average time (mm:ss)	SUS score (0–100)	Usability problems
WQS	**100**	**2,60**	**05:09**	**61**	4
QueryVOWL	95,56	4,27	07:44	50,5	7
SPARKLIS	93,33	4,53	07:21	48,5	**3**

Finally, as for **help and documentation**, users liked the example queries of WQS and SPARKLIS; and QueryVOWLs video. However, they would have preferred integrated tutorials over the available documentation of the tools.

4.3 Discussion

The results of the evaluation are summarized and discussed for each tool. An overview of the results, including the number of extracted usability problems, is shown in Table 5.

WQS. WQS had the best results regarding the usability metrics. Users were able to achieve the best effectiveness and efficiency by a wide margin. The analysis of think-aloud protocols revealed four usability problems: the use of confusing terms, missing user control, non-selectable options and the complexity of parts of the UI. The problem that most participants encountered was the clutter of parts of the UI, such as the suggestion box. Furthermore, buttons and input fields were described as hidden away and led to confusion among users. Overall, WQS had the best results in this user study, most likely, based on the easy to use form-based approach with recommendations and suggestions enabled by the Wikidata API. However, it offered the lowest amount of expressiveness and was the only tool that was domain-dependent.

QueryVOWL. In short, regarding both the amount of hints given and the time per task, the first task of QueryVOWL had the highest average of all tasks and tools. It was apparent that users had difficulties understanding the visual approach and the interface of the tool. The think-aloud protocols showed that both the frustration and difficulties that users experienced could be explained by the uncovered usability problems. Seven usability problems could be extracted from the protocols: the lack of appropriate feedback, use of confusing terms, missing user control, non-familiar visuals, missing error prevention, non-selectable options and the UI missing necessary information. One of the most substantial problems was the missing visibility of system status and appropriate feedback. This, combined with the use of confusing terms and the lack of error prevention, led to making mistakes. The absence of undo and redo functions only amplified these problems. However, it is important to note that the tested version of QueryVOWL was a prototype meant to demonstrate the querying approach.

SPARKLIS. Both the completion rate and the average total amount of hints were marginally worse than those of QueryVOWL. Three usability problems could be extracted from the think-aloud protocols: the lack of appropriate feedback, missing error prevention and the overwhelming UI. Most users were overwhelmed by the amount of options and input fields that SPARKLIS offered and suggested a more minimalistic interface. SPARKLIS results were interesting in that the usability metrics results differed from the statements and think-aloud protocols of users. Most users were satisfied with the tool, and users that were not, said that they liked its approach after seeing all tools. This combined with the low amount of usability problems speaks for the usability of the tool. The think-aloud protocols showed that a better tutorial or a beginner friendlier interface would have led to a more usable tool.

4.4 Lessons Learned

Based on our results and the similarities regarding usability problems and user suggestions, we summarize key aspects for designing a query builder for knowledge graphs.

First, results show that the querying approach is not as important for the usability of the tool as it may seem. User satisfaction and preference was mostly influenced by the interface design and ease of use of the tools. For example, 4 users preferred the graph-based approach of QueryVOWL even if they did not grade the tool itself as usable.

Regarding the ease of use, the availability of suggestions had a great impact on users. Casual users are inexperienced and suggestions allow them to see possible queries and subsequently understand the way queries are built. Participants suggested that tools could offer their most frequently built queries as examples, as (initial) queries built by casual users most likely would not differ too much.

An important point for the interface design was not to overwhelm the user with options. The results of think-aloud protocols show that most participants disliked having too many options or fields for query input. In contrast, important functionalities should not be hidden away from users. The features that are likely to be used often, should be visible and selectable at all times.

Regarding the usability heuristics, some features had a great impact on users. The ability to undo and redo parts of the query was praised when it was available and criticized when it was missing by a majority of participants. Casual users are especially prone to a trial and error method, which is why the possibility to undo errors as well as error prevention methods are so valuable.

Concerning the documentation, participants of the study said that they were not likely to read tutorials or watch videos longer than a few minutes. A majority of users suggested to integrate tutorials in the query builder interface. The availability of tooltips could also improve usability.

Finally, it should be noted that for most casual users the alternative to building a SPARQL query to gather information is the use of traditional keyword-based web search engines. A majority of participants said that if they had the choice they would still use those search engines instead of any query builder

to solve tasks such as those of the user study. Therefore, query builders still need to somehow compete with this traditional mindset, keeping high usability standards while offering advanced functionalities to exploit the expressivity of SPARQL and the rich fine-grained information of (potentially interconnected) knowledge graphs.

5 Conclusion and Future Work

This paper presents a usability evaluation of SPARQL query builders for laypeople, i.e., users that want to explore knowledge graphs but have no prior knowledge of SPARQL. We first categorize and analyze query builders based on their querying approach (i.e., form-, graph-, and natural language-based). We then select and evaluate three prominent representatives: the Wikidata Query Service (WQS), QueryVOWL and SPARKLIS.

Our user study is based on a mixed-method usability evaluation with three increasingly complex tasks (i.e. queries). On the one hand, we measure the effectiveness, efficiency and the System Usability Scale (SUS) score as quantitative data. On the other hand, we make use of the think-aloud method as qualitative data, clustering results based on usability heuristics.

The results show that the form-based WQS offered the best usability of the three selected tools. However, usability problems were found for all tools, mostly concerning the difficulty of understanding and efficiently performing the query building process. Irrespective of the querying approach, users were mostly influenced by the interface design and ease of use of the tools.

Finally, we extract key aspects for the interface design of future query builders. These include the availability of undo functions and error prevention methods as well as integrated tutorials, examples and suggestions to understand how queries are constructed for the underlying knowledge graphs.

Our future work considers to expand the user study with a broader spectrum of queries and users, and the application of the lesson learned to build the next generation of query builders for knowledge graphs.

Acknowledgements. This work is supported by the EU's Horizon 2020 research and innovation programme: grant 731601 (SPECIAL), the Austrian Research Promotion Agency's (FFG) program "ICT of the Future": grant 861213 (CitySPIN).

References

1. Attard, J., Orlandi, F., Auer, S.: Exconquer: lowering barriers to rdf and linked data re-use. Semant. Web **9**(2), 241–255 (2018)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
3. Bangor, A., Kortum, P., Miller, J.: Determining what individual sus scores mean: adding an adjective rating scale. J. Usability Stud. **4**(3), 114–123 (2009)

4. Bevan, N., Carter, J., Harker, S.: ISO 9241-11 revised: what have we learnt about usability since 1998? In: Kurosu, M. (ed.) HCI 2015. LNCS, vol. 9169, pp. 143–151. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20901-2_13
5. Bonatti, P.A., Decker, S., Polleres, A., Presutti, V.: Knowledge graphs: new directions for knowledge representation on the semantic web (Dagstuhl seminar 18371). Dagstuhl Rep. **8**(9), 29–111 (2019)
6. Brooke, J.: SUS: a retrospective. J. Usability Stud. **8**(2), 29–40 (2013)
7. Charters, E.: The use of think-aloud methods in qualitative research anintroduction to think-aloud methods. Brock Educ. J. OLD **12**(2) (2003)
8. Ferré, S.: SPARKLIS: an expressive query builder for sparql endpoints with guidance in natural language. Semant. Web **8**(3), 405–418 (2017)
9. Grafkin, P., Mironov, M., Fellmann, M., Lantow, B., Sandkuhl, K., Smirnov, A.V.: SPARQL query builders: overview and comparison. In: BIR Workshops (2016)
10. Greene, J.C., Caracelli, V.J., Graham, W.F.: Toward a conceptual framework for mixed-method evaluation designs. Educ. Eval. Policy Anal. **11**(3), 255–274 (1989)
11. Haag, F., Lohmann, S., Siek, S., Ertl, T.: QueryVOWL: visual composition of SPARQL queries. In: Gandon, F., Guéret, C., Villata, S., Breslin, J., Faron-Zucker, C., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9341, pp. 62–66. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25639-9_12
12. Harris, S., Seaborne, A.: SPARQL 1.1 query language, March 2013
13. Heath, T., Bizer, C.: Linked data: evolving the web into a global data space. Synth. Lect. Semant. Web: Theory Technol. **1**(1), 1–136 (2011)
14. Hoefler, P., Granitzer, M., Veas, E.E., Seifert, C.: Linked data query wizard: a novel interface for accessing SPARQL endpoints. In: Proceedings of LDOW (2014)
15. Kaufmann, E., Bernstein, A.: How useful are natural language interfaces to the semantic web for casual end-users? In: Aberer, K., et al. (eds.) ASWC/ISWC - 2007. LNCS, vol. 4825, pp. 281–294. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_21
16. Kaufmann, E., Bernstein, A., Fischer, L.: NLP-reduce: a naive but domain-independent natural language interface for querying ontologies. In: Proceedings of ESWC, pp. 1–2 (2007)
17. Lehmann, J., Bühmann, L.: AutoSPARQL: let users query your knowledge base. In: Antoniou, G., et al. (eds.) ESWC 2011. LNCS, vol. 6643, pp. 63–79. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21034-1_5
18. Lohmann, S., Negru, S., Haag, F., Ertl, T.: Visualizing ontologies with VOWL. Semant. Web **7**(4), 399–419 (2016)
19. Nielsen, J.: Enhancing the explanatory power of usability heuristics. In: Proceedings of the SIGCHI conference on Human Factors in Computing Systems, pp. 152–158. ACM (1994)
20. Nielsen, J., Landauer, T.K.: A mathematical model of the finding of usability problems. In: Proceedings of INTERACT and CHI, pp. 206–213. ACM (1993)
21. Smart, P.R., Russell, A., Braines, D., Kalfoglou, Y., Bao, J., Shadbolt, N.R.: A visual approach to semantic query design using a web-based graphical query designer. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 275–291. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87696-0_25
22. Soylu, A., et al.: OptiqueVQS: a visual query system over ontologies for industry. Semant. Web **9**(5), 627–660 (2018)
23. Vrandecic, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)